# 

# CSP3341: Programming Languages and Paradigms – Workshop 4

## Version D

**Marks:** Marked out of 28, (7% of unit) **Due:** 6pm Friday of Week 5

**Name:**

**Student #:**

## Workshop Description

This workshop revises and applies topics from Chapters 3 and 4 of the textbook. It consists of a number of small tests and exercises. The content of this workshop is similar to that which may be included in your exam. **This workshop is assessed**. This workshop is to be completed **individually**.

### Task 1 – Grammars, Sentential Forms and Parse Trees (8 marks – 2 marks per part)

**A.** Looking at the BNF grammar for “treadmill”, below, describe the structure of a program in the treadmill language, in English. You may assume “inst” is an instruction and “int\_lit” is an integer.

**English description of “treadmill”**

**Grammar for “treadmill”**

<program> **→ BEGIN** <inst\_list> **END**

<inst\_list> **→** <inst>

**|** <inst>**,** <inst\_list>

<inst> **→ SET INCLINE** <angle>

**| SET SPEED** <speed>

**| WAIT** int\_lit **SECONDS**

**| WAIT** int\_lit **MINUTES**

<angle> **→ 0 | 1 | 3 | 5 | 7 | 9 | 11**

<speed> **→ 1 | 2 | 3 | 4 | 5 | 6 | 7**

**B.** Identify which of the programs are not syntactically correct according to the grammar of “treadmill”. If a program is not valid, highlight the errors in the program.

|  |  |
| --- | --- |
| **Programs in “treadmill” Grammar** | **Valid?** |
| BEGIN SET SPEED 5, WAIT 5 MINUTES, END |  |
| BEGIN SET SPEED 2, SET INCLINE 5, WAIT 10 MINUTES, SET INCLINE 2 END |  |
| BEGIN SET INCLINE 3, SET SPEED 2, WAIT 30 SECONDS, SET INCLINE 7, SET SPEED 4, WAIT 5 MINUTES, SET INCLINE 0, SET SPEED 2, WAIT 2 MINUTES END |  |
| BEGIN SET INCLINE 1, SET INCLINE 3, SET INCLINE 5, SET INCLINE 7, SET INCLINE 9, SET INCLINE 11 END |  |
| BEGIN SET SPEED 5, WAIT 2 MINUTES, SET ANGLE 3, WAIT 1 MINUTE, SET SPEED 9, WAIT 30 SECONDS, SET ANGLE 0 END |  |
| BEGIN, SET SPEED 2, SET INCLINE 3, WAIT 1 MINUTES, SET INCLINE 7 SPEED, WAIT 35 SECONDS, SET SPEED 1, WAIT 1 MINUTE END |  |

**C.** Using **leftmost derivation** and the “treadmill” grammar, derive the statement “BEGIN SET SPEED 5, SET INCLINE 3, WAIT 5 MINUTES END” into its final sentential form. Show all sentential forms along the way.

**Derivation of “BEGIN SET SPEED 5, SET INCLINE 3, WAIT 5 MINUTES END”**

<program> => BEGIN <inst\_list> END

**D.** Construct a parse tree of “BEGIN SET SPEED 5, SET INCLINE 3, WAIT 5 MINUTES END” using the “treadmill” grammar. It should match your derivation from part B of this task.

**Parse Tree of “BEGIN SET SPEED 5, SET INCLINE 3, WAIT 5 MINUTES END”**

**BEGIN** <inst\_list> **END**

### Task 2 – Converting BNF to EBNF (4 marks)

Translate the “treadmill” grammar rules (Task 1) from BNF form to Extended BNF form, using the same notation to that used in the textbook. Your final EBNF grammar should describe the same language as the BNF grammar, and be highly readable.

**EBNF Grammar of “treadmill”**

### Task 3 – Axiomatic Semantics & Weakest Preconditions (6 marks)

Identify the weakest preconditions and postconditions for all statements in the sequence. Assume only integer values. Also identify the weakest precondition for the “if” statement. Write the weakest preconditions/postconditions inside the braces provided.

**Weakest Precondition – If**

{}

if ( a > 10 )

b = b + 6

else

b = b \* 7

{ **b < 15** }

**Weakest Precondition - Sequence**

{ }

x = x + 3 - y

{ }

y = 5 \* (15 - x)

{ }

x = 20 + y;

{ **x > 30** }

### Task 4 – Lexical Analysis (2 marks)

The table below is an abstraction of the tokens and character classes in a lexical analyser.

|  |  |
| --- | --- |
| **Tokens** | **Description** |
| VAR | LETTER, followed by one or more DIGIT or LETTER |
| INT | one or more DIGIT |
| FLOAT | one or more DIGIT, then DEC\_PNT, then one or more DIGIT |
| ADD\_OP | the + character |
| SUB\_OP | the - character |
| MULT\_OP | the \* character |
| DIV\_OP | the / character |
| ASSIGN\_OP | the = character |
| DEC\_PNT | the . character |
| L\_PAREN | the ( character |
| R\_PAREN | the ) character |
| SEMICOLON | the ; character |
|  |  |
| **Character Classes** | **Description** |
| LETTER | all upper and lower case letters ([A-Za-z]) |
| DIGIT | all numbers ([0-9]) |

Using the definitions in the table, simulate the output (to the syntax analyser) of the lexical analyser given the following string of input. You may assume (and omit) the presence of EOF after the input.

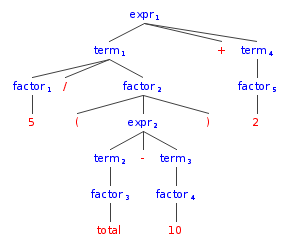
**bigSum = 2 \* (sum1 + offset) \* (5.5 - quota);**

|  |  |
| --- | --- |
| **Token** | **Lexeme** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Task 5 – Syntax Analysis (Recursive-Descent Parsing) (4 marks)

|  |
| --- |
| **NxtToken:** INT **NxtLexeme:** 5 |
| **Enter** expr |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Below is a parse tree of “5 / (total - 10) + 2”, using the same grammar that is used in the textbook’s section on Recursive-Descent Parsing (4.4.1). Complete the table next to the tree, illustrating a trace of the parser’s progress and use of expr, term and factor functions.



Tree generated in phpSyntaxTree - <http://ironcreek.net/phpsyntaxtree/>

Grammar from Section 4.4.1 of textbook:

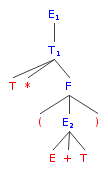
<expr> → <term> {(**+** | **-**) <term>}

<term> → <factor> {(**\*** | **/**) <factor>}

<factor> → **id** | **int\_constant** | **(** <expr> **)**

### Task 6 – Syntax Analysis (Bottom-Up Parsing), Phrases and Handles (4 marks)

Below left are a simple bottom-up grammar and a parse tree of a right sentential form of a derivation using the grammar. Identify and list the phrases, simple phrases and handle.



**Bottom-Up Grammar**

E **→** E **+** T **|** T

T **→** T **\*** F **|** F

F **→ (**E**)** **| id**

Right sentential form:

**T \* (E + T)**

**Handle**

Place a around the handle

**Simple Phrases**

**Phrases**